

What is claimed is.

- 1 1. A method for validating a processor design by
2 simulating program execution, comprising the steps of:
3 identifying a resource that may be accessed by a test
4 program that includes a first simulated process and a
5 second simulated process;
6 associating a set of non-unique values for said re-
7 source;
8 executing said test program by the steps of:
9 executing a first sequence of instructions in said
10 first simulated process; and
11 while performing said step of executing said first
12 sequence, executing a second sequence of instructions in
13 said second simulated process;
14 wherein said resource is accessed by at least one of
15 said first simulated process and said second simulated
16 process, and wherein upon completion of said steps of
17 executing said first sequence and executing said second
18 sequence, a member of said set of non-unique values is
19 required to be present in said resource; and
20 verifying an equality between a content of said re-
21 source and a member of said set of non-unique values.

- 1 2. The method according to claim 1, wherein said re-
2 source is a memory resource.

1 3. The method according to claim 1, wherein said re-
2 source is a register.

1 4. The method according to claim 1, wherein said set
2 of non-unique values is a set of value-lists.

1 5. The method according to claim 4, wherein said re-
2 source comprises a first adjacent resource and a second
3 adjacent resource, and each member of said set of
4 value-lists comprises a first value and a second value,
5 said first value being a permissible value of said first
6 adjacent resource, and said second value being a permis-
7 sible values of said second adjacent resource, and said
8 step of verifying further comprising the steps of:

9 identifying a valid member of said set of
10 value-lists, by the steps of:

11 verifying an equality between a content of said first
12 adjacent resource and said first value of said valid mem-
13 ber; and

14 verifying an equality between a content of said sec-
15 ond adjacent resource and said second value of said valid
16 member.

1 6. The method according to claim 1, wherein said re-
2 source comprises a first resource and a second resource
3 and said set of non-unique values comprises a first set
4 of non-unique values that is associated with said first
5 resource, and a second set of non-unique values that is
6 associated with said second resource, further comprising
7 the steps of:

8 associating a first member of said first set of
9 non-unique values with a second member of said second set
10 of non-unique values; and

11 wherein said step of verifying comprises the steps
12 of:

13 verifying an equality between said first resource and
14 said first member; and

15 verifying an equality between said second resource
16 and said second member.

1 7. The method according to claim 6, wherein said step
2 of associating said first member is performed by tagging
3 said first member and said second member with a common
4 combination identifier.

1 8. The method according to claim 6, said first member
2 of said first set of non-unique values comprises a first
3 value-list, and said second member of said second set of
4 non-unique values comprises a second value-list, respec-
5 tive elements of said first value-list being permissible
6 values of said first resource and adjacent resources
7 thereof, and respective elements of said second
8 value-list being permissible values of said second re-
9 source and adjacent resources thereof,

10 wherein said step of verifying comprises the steps of
11 verifying an equality between a content of said first
12 resource and adjacent resources thereof with correspond-
13 ing elements of said first value-list; and
14 verifying an equality between a content of said sec-
15 ond resource and adjacent resources thereof with corre-
16 sponding elements of said second value-list.

1 9. The method according to claim 1, wherein said step
2 of associating said set of non-unique values is performed
3 prior to said step of executing said first sequence of
4 instructions.

1 10. The method according to claim 9, further compris-
2 ing the steps of:

3 defining a results section in said test program, and
4 entering all permissible values assumable by said re-
5 source and an identifier of said resource in an entry of
6 said results section.

1 11. The method according to claim 1, wherein said
2 step of executing said test program further comprises the
3 steps of:

4 generating said first sequence and said second se-
5 quence to define generated instructions;

6 while performing said step of generating, simulating
7 one of said generated instructions in said first simu-
8 lated process and said second simulated process;

9 maintaining a store that contains a set of values
10 that are assumable in said resource during said step of
11 simulating said one of said generated instructions; and

12 thereafter determining whether said store contains
13 non-unique values.

1 12. The method according to claim 11, further com-
2 prising the steps of:

3 storing in said store first values of said resource
4 during accesses thereof by said first simulated process;
5 and

6 storing in said store second values of said resource
7 by said second simulated process during accesses thereof,
8 wherein said first values comprise first written values,
9 and said second values comprise second written values;
10 and;

11 identifying in said store a last value written to
12 said resource in said step of simulating said one of said
13 generated instructions.

1 13. The method according to claim 11,
2 wherein said step of maintaining said store further
3 comprises the steps of:
4 determining an initial state of said test program im-
5 mediately prior to performing said step of simulating
6 said one of said generated instructions;
7 storing in said store written values of target re-
8 sources of said one generated instruction during accesses
9 thereof; and
10 further comprising the steps of:
11 identifying all combinations of values of input re-
12 sources of said one generated instruction;
13 rolling back said test program to reestablish said
14 initial state;
15 thereafter resimulating said one of said generated
16 instructions using each of said combinations of values;
17 reidentifying written values that are written to said
18 target resources; and
19 updating said store with reidentified written values
20 of said target resources.

1 14. The method according to claim 1, further compris-
2 ing the step of establishing a synchronization barrier
3 for said first simulated process and said second simu-
4 lated process.

1 15. The method according to claim 1, further comprising
2 the steps of biasing generation of said test program
3 to promote collisions of memory accessing instructions
4 that are executed by said first simulated process and
5 said second simulated process.

1 16. A method of verification of an architecture by
2 simulation, comprising the steps of:

3 defining a program input to a test generator;
4 generating a test program responsive to said program
5 input said test program including a list of resource ini-
6 tializations, a list of instructions, and a list of pre-
7 dicted resource results, wherein at least one member of
8 said list of predicted resource results comprises a plu-
9 rality of permissible results;

10 simulating an execution of said test program using a
11 plurality of simultaneously executing processes; and

12 verifying an actual resource result by determining
13 that at least one of said plurality of permissible re-
14 sults is equal to said actual resource result.

1 17. The method according to claim 16, wherein said
2 list of predicted resource results comprises predicted
3 results of adjacent resources, wherein said adjacent re-
4 sources are mutually dependent, and said step of verify-
5 ing said actual resource result is performed by verifying
6 each of said adjacent resources.

1 18. The method according to claim 17, wherein said
2 adjacent resources are memory resources.

1 19. The method according to claim 17, wherein said
2 adjacent resources are registers.

1 20. The method according to claim 16, wherein said
2 list of predicted resource results comprises predicted
3 results of mutually dependent non-adjacent resources,
4 further comprising the steps of:

5 identifying a combination of said mutually dependent
6 non-adjacent resources by tagging corresponding members
7 of said list of predicted resource results with a unique
8 combination identifier to define commonly tagged
9 value-lists of predicted resource results; and

10 said step of verifying said actual resource result is
11 performed by verifying that resources of said combination
12 have actual results that are equal to a member of a cor-
13 responding one of said commonly tagged value-lists.

1 21. A method of predicting non-unique results by
2 simulating a system design, comprising the steps of:

3 defining a program input to a test generator;

4 generating a test program responsive to said program
5 input, said test program including a list of resource
6 initializations, a list of instructions, and a list of
7 predicted resource results, wherein at least one member

8 of said list of predicted resource results comprises a
9 plurality of permissible results;

10 simulating an execution of a single instruction of
11 said test program by a first process of said test pro-
12 gram; and

13 calculating possible values of target resources of
14 said single instruction.

1 22. The method according to claim 21, wherein said
2 target resources are memory resources.

1 23. The method according to claim 21, wherein said
2 target resources are registers.

1 24. The method according to claim 21, further com-
2 prising the steps of:

3 performing said step of simulating an execution by a
4 second process of said test program;

5 maintaining process-linked lists of written values
6 that are written to said target resources of said single
7 instruction by said first process and said second proc-
8 ess; and

9 determining respective last values in said proc-
10 ess-linked lists of written values.

1 25. The method according to claim 16, further com-
2 prising the step of establishing a synchronization bar-
3 rier for said simultaneously executing processes.

1 26. The method according to claim 16, further comprising the steps of biasing generation of said test program to promote collisions of memory accessing instructions that are executed by said simultaneously executing processes.

1 27. A computer software product, comprising a computer-readable medium in which computer program instructions are stored, which instructions, when read by a computer, cause the computer to execute a method for validating a processor design by simulating program execution, comprising the steps of:

7 identifying a resource that may be accessed by a test program that includes a first simulated process and a second simulated process;

10 associating a set of non-unique values for said resource;

12 executing said test program by the steps of:

13 executing a first sequence of instructions in said first simulated process; and

15 while performing said step of executing said first sequence, executing a second sequence of instructions in said second simulated process;

18 wherein said resource is accessed by at least one of said first simulated process and said second simulated process, and wherein upon completion of said steps of executing said first sequence and executing said second

22 sequence, a member of said set of non-unique values is
23 required to be present in said resource; and
24 verifying an equality between a content of said re-
25 source and a member of said set of non-unique values.

1 28. The computer software product according to
2 claim 27, wherein said resource is a memory resource.

1 29. The computer software product according to
2 claim 27, wherein said resource is a register.

1 30. The computer software product according to claim
2 27, wherein said set of non-unique values is a set of
3 value-lists.

1 31. The computer software product according to
2 claim 30, wherein said resource comprises a first adja-
3 cent resource and a second adjacent resource, and each
4 member of said set of value-lists comprises a first value
5 and a second value, said first value being a permissible
6 value of said first adjacent resource, and said second
7 value being a permissible values of said second adjacent
8 resource, and said step of verifying further comprising
9 the steps of:

10 identifying a valid member of said set of
11 value-lists, by the steps of:

12 verifying an equality between a content of said first
13 adjacent resource and said first value of said valid mem-
14 ber; and

15 verifying an equality between a content of said sec-
16 ond adjacent resource and said second value of said valid
17 member.

1 32. The computer software product according to
2 claim 27, wherein said resource comprises a first re-
3 source and a second resource and said set of non-unique
4 values comprises a first set of non-unique values that is
5 associated with said first resource, and a second set of
6 non-unique values that is associated with said second re-
7 source, the method further comprising the steps of:

8 associating a first member of said first set of
9 non-unique values with a second member of said second set
10 of non-unique values; and

11 wherein said step of verifying comprises the steps
12 of:

13 verifying an equality between said first resource and
14 said first member; and

15 verifying an equality between said second resource
16 and said second member.

1 33. The computer software product according to
2 claim 32, wherein said step of associating said first
3 member is performed by tagging said first member and said
4 second member with a common combination identifier.

1 34. The computer software product according to
2 claim 32, said first member of said first set of
3 non-unique values comprises a first value-list, and said
4 second member of said second set of non-unique values
5 comprises a second value-list, respective elements of
6 said first value-list being permissible values of said
7 first resource and adjacent resources thereof, and re-
8 spective elements of said second value-list being permis-
9 sible values of said second resource and adjacent re-
10 sources thereof,

11 wherein said step of verifying comprises the steps of
12 verifying an equality between a content of said first
13 resource and adjacent resources thereof with correspond-
14 ing elements of said first value-list; and

15 verifying an equality between a content of said sec-
16 ond resource and adjacent resources thereof with corre-
17 sponding elements of said second value-list.

1 35. The computer software product according to
2 claim 27, wherein said step of associating said set of
3 non-unique values is performed prior to said step of exe-
4 cuting said first sequence of instructions.

1 36. The computer software product according to
2 claim 35, wherein said step of associating said set of
3 non-unique values is performed by the steps of:

4 defining a results section in said input statements,
5 and

6 entering all permissible values assumable by said re-
7 source and an identifier of said resource in an entry of
8 said results section.

1 37. The computer software product according to
2 claim 27, wherein said step of executing said test pro-
3 gram further comprises the steps of:

4 generating said first sequence and said second se-
5 quence to define generated instructions;

6 while performing said step of generating, simulating
7 one of said generated instructions in said first simu-
8 lated process and said second simulated process;

9 maintaining a store that contains a set of values
10 that are assumable in said resource during said step of
11 simulating said one of said generated instructions; and

12 thereafter determining whether said store contains
13 non-unique values.

1 38. The computer software product according to
2 claim 37, wherein said step of maintaining said store
3 further comprises the steps of:

4 maintaining a first store that contains first values
5 contained in said resource during accesses thereof by
6 said first simulated process; and

7 maintaining a second store that contains second val-
8 ues contained in said resource by said second simulated

9 process during accesses thereof, wherein said first val-
10 ues comprise first read values and first written values,
11 and said second values comprise second read values and
12 second written values; and

13 said step of determining whether said store contains
14 non-unique values further comprises the step of identify-
15 ing in one of said first store and said second store a
16 last value written to said resource in said step of simu-
17 lating said one of said generated instructions.

1 39. The computer software product according to
2 claim 38, the method further comprising the steps of:

3 identifying a first unique combination of said first
4 simulated process, and said first read values;

5 identifying a second unique combination of said sec-
6 ond simulated process, and said second read values; and

7 resimulating said one of said generated instructions
8 using a selected value that is selected from said first
9 read values of said first unique combination and said
10 second read values of said second unique combination as
11 an input to said one of said generated instructions; and
12 reidentifying values that are written to said re-
13 source in said step of resimulating.

1 40. The computer software product according to claim
2 27, further comprising the step of establishing a syn-
3 chronization barrier for said first simulated process and
4 said second simulated process.

1 41. The computer software product according to claim
2 27, further comprising the step of biasing generation of
3 said test program to promote collisions of memory access-
4 ing instructions that are executed by said first simu-
5 lated process and said second simulated process.

1 42. A computer software product, comprising a com-
2 puter-readable medium in which computer program instruc-
3 tions are stored, which instructions, when read by a com-
4 puter, cause the computer to perform a method of verifi-
5 cation of an architecture by simulation, comprising the
6 steps of:

7 defining a program input to a test generator;

8 generating a test program responsive to said program
9 input, said test program including a list of resource
10 initializations, a list of instructions, and a list of
11 predicted resource results, wherein at least one member
12 of said list of predicted resource results comprises a
13 plurality of permissible results;

14 simulating an execution of said test program using a
15 plurality of simultaneously executing processes; and

16 verifying an actual resource result by determining
17 that at least one of said plurality of permissible re-
18 sults is equal to said actual resource result.

1 43. The computer software product according to
2 claim 42, wherein said list of predicted resource results
3 comprises predicted results of adjacent resources,
4 wherein said adjacent resources are mutually dependent,
5 and said step of verifying said actual resource result is
6 performed by verifying each of said adjacent resources.

1 44. The computer software product according to
2 claim 43, wherein said adjacent resources are memory re-
3 sources.

1 45. The computer software product according to
2 claim 43, wherein said adjacent resources are registers.

1 46. The computer software product according to claim
2 42, further comprising the step of establishing a syn-
3 chronization barrier for said simultaneously executing
4 processes.

1 47. The computer software product according to claim
2 42, further comprising the steps of biasing generation of
3 said test program to promote collisions of memory access-
4 ing instructions that are executed by said simultaneously
5 executing processes.

1 48. The computer software product according to
2 claim 42, wherein said list of predicted resource results
3 comprises predicted results of mutually dependent
4 non-adjacent resources, the method further comprising the
5 steps of:

6 identifying a combination of said mutually dependent
7 non-adjacent resources by tagging corresponding members
8 of said list of predicted resource results with a unique
9 combination identifier to define commonly tagged
10 value-lists of predicted resource results; and

11 said step of verifying said actual resource result is
12 performed by verifying that resources of said combination
13 have actual results that are equal to a member of a cor-
14 responding one of said commonly tagged value-lists.

1 49. A computer software product, comprising a com-
2 puter-readable medium in which computer program instruc-
3 tions are stored, which instructions, when read by a com-
4 puter, cause the computer to perform a method of predict-
5 ing non-unique results by simulating a system design,
6 comprising the steps of:

7 defining a program input to a test generator;
8 generating a test program responsive to said program
9 input, said test program including a list of resource
10 initializations, a list of instructions, and a list of
11 predicted resource results, wherein at least one member
12 of said list of predicted resource results comprises a
13 plurality of permissible results;

14 simulating an execution of a single instruction of
15 said test program by a first process of said test pro-
16 gram; and
17 calculating possible values of target resources of
18 said single instruction.

1 50. The computer software product according to
2 claim 49, the method further comprising the steps of:
3 performing said step of simulating an execution by a
4 second process of said test program;
5 maintaining process-linked lists of written values
6 that are written to said target resources of said single
7 instruction by said first process and said second proc-
8 ess; and
9 determining respective last values in said proc-
10 cess-linked lists of written values.

1 51. The computer software product according to
2 claim 50, the method further comprising the steps of:
3 prior to performing said steps of simulating said
4 execution by said first process and of simulating said
5 execution by said second process memorizing an initial
6 simulated state of said test program;
7 maintaining process-linked lists of read values that
8 are read from source resources of said single instruc-
9 tion;
10 identifying a member of said lists of read values
11 having a unique value to define an identified member;

12 restoring said initial simulated state of said test
13 program; and

14 performing said step of simulating said execution a
15 second time, reading said identified member, using an as-
16 sociated process thereof.

1 52. An apparatus for verifying a design comprising a
2 simulation processor which is adapted to perform the
3 steps of:

4 identifying a resource that may be accessed by a test
5 program that includes a first simulated process and a
6 second simulated process;

7 associating a set of non-unique values for said re-
8 source;

9 executing said test program by the steps of:

10 executing a first sequence of instructions in said
11 first simulated process; and

12 while performing said step of executing said first
13 sequence, executing a second sequence of instructions in
14 said second simulated process;

15 wherein said resource is accessed by at least one of
16 said first simulated process and said second simulated
17 process, and wherein upon completion of said steps of
18 executing said first sequence and executing said second
19 sequence, a member of said set of non-unique values is
20 required to be present in said resource; and

21 verifying an equality between a content of said re-
22 source and a member of said set of non-unique values.

1 53. The apparatus according to claim 52, wherein said
2 resource is a memory resource.

1 54. The apparatus according to claim 52, wherein said
2 resource is a register.

1 55. The apparatus according to claim 52, wherein said
2 resource comprises a first adjacent resource and a second
3 adjacent resource, and said set of non-unique values com-
4 prises a first subset of non-unique values and a second
5 subset of non-unique values, said first subset of
6 non-unique values being permissible values of said first
7 adjacent resource, and said second subset of non-unique
8 values being permissible values of said second adjacent
9 resource, wherein said processor is further adapted to
10 perform the steps of:

11 establishing at least one allowable subcombination of
12 members of said set of non-unique values, a first member
13 of said subcombination being selected from said first
14 subset of non-unique values and a second member of said
15 subcombination being selected from said second subset of
16 non-unique values; and

17 wherein said step of verifying is performed by the
18 steps of:

19 verifying an equality between a content of said first
20 adjacent resource and said first member of said subcombi-
21 nation; and

22 verifying an equality between a content of said sec-
23 ond adjacent resource and said second member of said sub-
24 combination.

1 56. The apparatus according to claim 52, wherein said
2 resource comprises a first resource and a second resource
3 and said set of non-unique values comprises a first set
4 of non-unique values that is associated with said first
5 resource, and a second set of non-unique values that is
6 associated with said second resource, wherein said proc-
7 essor is further adapted to perform the steps of:

8 associating a first member of said first set of
9 non-unique values with a second member of said second set
10 of non-unique values; and

11 wherein said step of verifying comprises the steps
12 of:

13 verifying an equality between said first resource and
14 said first member; and

15 verifying an equality between said second resource
16 and said second member.

1 57. The apparatus according to claim 56, wherein said
2 step of associating said first member is performed by
3 tagging said first member and said second member with a
4 common combination identifier.

1 58. The apparatus according to claim 56, said first
2 member of said first set of non-unique values comprises a
3 first value-list, and said second member of said second
4 set of non-unique values comprises a second value-list,
5 respective elements of said first value-list being per-
6 missible values of said first resource and adjacent re-
7 sources thereof, and respective elements of said second
8 value-list being permissible values of said second re-
9 source and adjacent resources thereof,

10 wherein said step of verifying comprises the steps of
11 verifying an equality between a content of said first
12 resource and adjacent resources thereof with correspond-
13 ing elements of said first value-list; and

14 verifying an equality between a content of said sec-
15 ond resource and adjacent resources thereof with corre-
16 sponding elements of said second value-list.

1 59. The apparatus according to claim 52, wherein said
2 step of associating said set of non-unique values is per-
3 formed prior to said step of executing said first se-
4 quence of instructions.

1 60. The apparatus according to claim 59, wherein said
2 step of associating said set of non-unique values is per-
3 formed by the steps of:

4 providing input statements to a test generator; and
5 defining a results section in said input statements,
6 and

7 entering all permissible values assumable by said re-
8 source and an identifier of said resource in an entry of
9 said results section.

1 61. The apparatus according to claim 52, wherein said
2 step of executing said test program further comprises the
3 steps of:

4 generating said first sequence and said second se-
5 quence to define generated instructions;

6 while performing said step of generating, simulating
7 one of said generated instructions in said first simu-
8 lated process and said second simulated process;

9 maintaining a store that contains a set of values
10 that are assumable in said resource during said step of
11 simulating said one of said generated instructions; and

12 thereafter determining whether said store contains
13 non-unique values.

1 62. The apparatus according to claim 61, wherein said
2 step of maintaining said store further comprises the
3 steps of:

4 maintaining a first store that contains first values
5 contained in said resource during accesses thereof by
6 said first simulated process; and

7 maintaining a second store that contains second val-
8 ues contained in said resource by said second simulated
9 process during accesses thereof, wherein said first val-
10 ues comprise first read values and first written values,

11 and said second values comprise second read values and
12 second written values; and

13 said step of determining whether said store contains
14 non-unique values further comprises the step of identify-
15 ing in one of said first store and said second store a
16 last value written to said resource in said step of simu-
17 lating said one of said generated instructions.

1 63. The apparatus according to claim 62, wherein said
2 processor is further adapted to perform the steps of:

3 identifying a first unique combination of said first
4 simulated process, and said first read values;

5 identifying a second unique combination of said sec-
6 ond simulated process, and said second read values; and

7 resimulating said one of said generated instructions
8 using a selected value that is selected from said first
9 read values of said first unique combination and said
10 second read values of said second unique combination as
11 an input to said one of said generated instructions; and
12 reidentifying values that are written to said re-
13 source in said step of resimulating.

1 64. The apparatus according to claim 63, wherein said
2 processor is further adapted to perform the steps of de-
3 termining an initial state of said test program immedi-
4 ately prior to performing said step of simulating said
5 one of said generated instructions; and

6 rolling back said test program to reestablish said
7 initial state prior to performing said step of resimulat-
8 ing.

1 65. An apparatus for verifying an architecture by
2 simulation comprising a simulation processor which is
3 adapted to perform the steps of:

4 defining a program input to a test generator;
5 generating a test program responsive to said program
6 input, said test program including a list of resource
7 initializations, a list of instructions, and a list of
8 predicted resource results, wherein at least one member
9 of said list of predicted resource results comprises a
10 plurality of permissible results;

11 simulating an execution of said test program using a
12 plurality of simultaneously executing processes; and

13 verifying an actual resource result by determining
14 that at least one of said plurality of permissible re-
15 sults is equal to said actual resource result.

1 66. The apparatus according to claim 65, wherein said
2 list of predicted resource results comprises predicted
3 results of adjacent resources, wherein said adjacent re-
4 sources are mutually dependent, and said step of verify-
5 ing said actual resource result is performed by verifying
6 each of said adjacent resources.

1 67. The apparatus according to claim 65, wherein said
2 list of predicted resource results comprises predicted
3 results of mutually dependent non-adjacent resources,
4 wherein said processor is further adapted to perform the
5 steps of:

6 identifying a combination of said mutually dependent
7 non-adjacent resources by tagging corresponding members
8 of said list of predicted resource results with a unique
9 combination identifier to define commonly tagged
10 value-lists of predicted resource results; and

11 said step of verifying said actual resource result is
12 performed by verifying that resources of said combination
13 have actual results that are equal to a member of a cor-
14 responding one of said commonly tagged value-lists.

1 68. An apparatus for design verification in which
2 program instructions are stored, which instructions cause
3 a processor to execute a method of predicting non-unique
4 results by simulating a system design, comprising the
5 steps of:

6 defining a program input to a test generator;
7 generating a test program responsive to said program
8 input, said test program including a list of resource
9 initializations, a list of instructions, and a list of
10 predicted resource results, wherein at least one member
11 of said list of predicted resource results comprises a
12 plurality of permissible results;

13 simulating an execution of a single instruction of
14 said test program by a first process of said test pro-
15 gram; and

16 calculating possible values of target resources of
17 said single instruction.

1 69. The apparatus according to claim 68, wherein said
2 instructions cause the processor to further perform the
3 steps of:

4 performing said step of simulating an execution by a
5 second process of said test program;

6 maintaining process-linked lists of written values
7 that are written to said target resources of said single
8 instruction by said first process and said second proc-
9 ess; and

10 determining respective last values in said proc-
11 cess-linked lists of written values.

1 70. The apparatus according to claim 69, wherein said
2 instructions cause the processor to further perform the
3 steps of:

4 prior to performing said steps of simulating said
5 execution by said first process and of simulating said
6 execution by said second process memorizing an initial
7 simulated state of said test program;

8 maintaining process-linked lists of read values that
9 are read from source resources of said single instruc-
10 tion;

100

11 identifying a member of said lists of read values
12 having a unique value to define an identified member;
13 restoring said initial simulated state of said test
14 program; and
15 performing said step of simulating said execution a
16 second time, reading said identified member, using an as-
17 sociated process thereof.